GLUC User Guide

Jeffrey Terstriep
James Westervelt

October 6, 2004

GLUC is part of the Landuse Evolution and impact Assessment Model (LEAM) suite of software tools.

Overview

Purpose

GLUC is a land use evolution model that is useful for predicting the urban growth patterns 10-50 years into the future across a large area based on alternative local, county, state, and federal land use investments and policies. Examples include:

- Location and size of proposed county roads, state and federal highways, and size and access points (on-ramps) for limited-access (e.g. interstate) highways.
- Zoning
- Purchases of property (or property rights) to limit development.
- Construction of lakes and reservoirs.
- Establishment of permanent natural areas.
- Location of new employment centers.

The output of GLUC is a time-series of land use patterns that are predicted to develop based on the initial land use pattern, the land use investments and policies, and the projected population change.

GLUC projects three land use changes:

- Undeveloped to low-density urban
- Undeveloped to high-density urban
- Undeveloped to permanent open

GLUC is a spatially explicit raster-GIS based dynamic simulation model. It works with a spatial resolution of 30 meters (30-meter raster GIS grid cells) and uses a time step of one year. At each time step a hedonic modeling approach is used to identify the relative attractiveness (or value) of each grid cell for conversion from undeveloped to each of the three developed uses and uses a probability function to select cells that will change to meet the predefined population increases.

Caveats

Only the three land use changes are predicted meaning that GLUC will not address questions of urban decline in a city or a region. Although different socioeconomic classes are attracted to develop in different areas, GLUC does not directly accommodate such differences. Through calibration of the model relative overall attractiveness for development across the entire local population can be adjusted, but GLUC will not grow nor distinguish between different types of neighborhoods. Demand for transportation is not calculated during GLUC simulation runs nor are new transportation routes established (air, water, rail, or road). Planners are encouraged to use GLUC to test for the relative consequences of such plans. While road and highway transportation routes are considered, rail and water commuter routes are not. Also, road networks cannot be phased-in. Finally, GLUC is not optimized to predict detailed growth for a city, but rather general growth of a city within the context of its surrounding counties.

GLUC is designed to provide quick and reasonable urban growth projections in the 10-50 year time frame. It does not take into account the effects of current investments in land for development purposes and the activities of investors to encourage local municipalities to provide the road and utility infrastructure to optimize such investments. Therefore, the predictions of land development within the next 10 years are probably not optimal. GLUC computes relative attractiveness for land conversion based on a one-time planned road/highway network and location and size of employment centers (e.g. city centers). GLUC does not predict nor does it accommodate a phasing in of transportation networks or business centers. Therefore, GLUC results after several decades (e.g. 50 years) may not be useful.

User Requirements

GLUC is written in the C programming language and was originally developed for Unix systems. Its inputs are from raster GIS data. Therefore, GLUC model users must be familiar with:

1. Unix
2. Compilation of Unix software
3. Process for finding and retrieving software source code via the Internet
4. Raster GIS data import, export, and analysis

In addition, large models (e.g. 20 million active raster grid cells or more) are best run on high performance computing systems. GLUC has been ported and is installed on a number of super computers at the Major Shared Resource Center (MSRC) site in Vicksburg, Mississippi and the Naval Oceanographic Laboratory at the Stennis Space Center. Users of these site must be familiar with running software on these computers, and have secure logins to them.

History

GLUC is being developed as part of the LEAM (Landuse Evolution and Impact Assessment Model) suite of software and procedures. LEAM was initially developed with an NSF grant to the Departments of Urban and Regional Planning and Landscape Architecture at the University of Illinois at Urbana-Champaign. Dr. Brian Deal directed the development of LEAM as part of his doctoral dissertation and now directs the LEAM lab (http://www.LEAM.uiuc.edu). LEAM has been successfully been used to evaluate and report urbanizing patterns around a number of cities as well as social, economic, and ecologic consequences of the growth. LEAM is a process that involves the development and operation of a landuse evolution model, evaluation of projected landuse patterns using a GIS, and posting of the results through interactive Web-based interfaces.

GLUC development focuses the GLUC landuse evolution modeling on the problem of urban encroachment near military installations. The landuse projections can be evaluated with respect to both impacts on existing or planned use of military installation training and testing areas and loss of long-term installation training and testing opportunities. These analyses are not covered in this document.

Software Requirements

GLUC

GLUC is a publicly available software code written in the C programming language.

GIS

GLUC is raster-GIS based and the GLUC modeler may choose from any of a number of systems – including the public-domain GRASS system. Using the raster GIS, modelers must process nationally available GIS data to create the input maps required to run GLUC.

Data

GLUC has been developed to allow for basic models to be created using raster GIS data readily available for anywhere in the United States. Local data – especially information about scenarios to be tested can augment the development of the GLUC input maps.

Web site

GLUC is part of the Army Corps of Engineers' Sustainability, Encroachment, and Room to Maneuver (SERM) efforts, which can be found at

http://www.cecer.army.mil/KD/SERM

This site contains:

1. The GLUC software
2. GLUC binaries for high performance supercomputers
3. Models developed for military installations
4. Manuals and other documentation
5. User feedback opportunities

To prepare to use GLUC, potential users must become familiar with the following:

1. UNIX
2. Raster GIS operations
3. GLUC

Modeling Steps

Overview

The successful application of GLUC to challenges related to military installation encroachment requires the commitment of a multidisciplinary team that can successfully work with a multi-stakeholder community, and run cutting-edge Unix-based software. The following table outlines an ambitious schedule that can result in the generation of useful feedback to stakeholders on the potential long-term implications of proposed land management investments and policies.

Activity	**Estimated FTE Time**
Coordinate with local communities	1 to 2 weeks
Acquire and compile GLUC	1 day
Acquire data	Days weeks
Prepare GLUC input maps	1 week
Setup GLUC	1 day
Calibrate	
Run GLUC	1 day to 2 weeks
Review Results	1 day to 1 week

Table 1: Modeling steps

Each of these steps if developed in detail below.

Coordinate with Local Community

GLUC helps answer the question "If we invest in these regional infrastructure projects, support this zoning plan, and purchase properties for these purposes, how will urban development proceed across the landscape 10-50 years form now?" Local communities and municipalities involve many stakeholders that carry dreams of what the region should look like in the future and have ideas of how to achieve those dreams. The primary purpose of GLUC is to test these ideas with respect to the potential for achieving desired future land use patterns. Therefore any useful GLUC simulation runs must begin and end with community participation. It begins with extracting an understanding of competing proposals for land planning investments and policies and ends with comparisons of how well the proposals can achieve the desired future state.

Acquire and compile GLUC

GIS

GLUC is fundamentally raster GIS based. A raster GIS will be required to prepare, manipulate, and display results. A GRASS script (in the form of a Unix makefile) is available to prepare the GLUC input maps. Potential users are encouraged to use

Acquire and Install GLUC

The GLUC source code can be downloaded from http://www.cecer.army.mil/kd/serm. Instructions are available with the source for compiling on a variety of machines.

Acquire Data

GLUC was developed to allow for inexpensive analyses of the impact of proposed regional investments and policies on future urban patterns. As such, it was designed to depend on nationally available data sets. The information below will help a modeler find the base data that will support GLUC analyses.

GIS

GLUC is raster GIS based and was designed to make use of nationally available data. GLUC, however does not make direct use of this data; instead, the data must be meticulously processed using your raster GIS. Therefore, the actual data used to create the required GLUC input files can be of the users choosing. However, the following sources for data maps are recommended:

Land Cover Maps

Land Cover maps can be downloaded from the USGS (http://seamless.usgs.gov/). Pick a rectangular region of interest using a mouse and download. The maps are in a TIFF file format and provide land cover using the National Land Cover Data (NLCD) classification system (http://landcover.usgs.gov/classes.html). The maps are available at 1:24,000 scale and 30-meter spatial resolution.

Digital Elevation Maps

Seamless national DEM maps are also available from the USGS (http://seamless.usgs.gov/). Oncedownloaded, each zipped file becomes a grid by unzipping. Data is at 1:24000 scale and 30-meter spatial resolution.

Boundary Maps

County and incorporated places (municipal) maps are required. County boundaries are available from the U.S. Census Bureau (http://www.census.gov/geo/www/cob/co2000.html). They are sorted by FIPS code. Incorporated places data are available for download at http://arcdata.esri.com/data/tiger2000/tiger_download.cfm. In addition population data for urban areas is required and available from the website.

Road Network Map

Road networks are available from the U.S. Census Bureau and can be downloaded at http://arcdata.esri.com/data/tiger2000/tiger_download.cfm. Included with these maps is a classification code (CFCC) designating limited-access hwy (A1), US route (A2), state route (A3) and roads and streets (A4). A scale of 1:24,000 is preferred. Ramp information (A6) for limited-access roads is also included.

Ownership Property Map

These areas represent areas that may be unavailable for development because of long-term government control for example Parks, Nature Preserves, Military Bases, Federal Lands, and Indian Reservations. This information is available from a variety of organizations including USGS, DoD, BLM, etc. Landmark data from http://arcdata.esri.com/data/tiger2000/tiger_download.cfm covers most of undevelopable lands.

Floodplain Map

Development is restricted from floodplains. Flood Hazard Boundary Maps are limitedly available from FEMA (http://web1.msc.fema.gov/). Some local agencies have 100yr and 500yr flood zone data (e.g. http://www.isgs.uiuc.edu/nsdihome/webdocs/county.html for Illinois flood zone by county)

Social

Population

GLUC urban growth predictions are based on projected populations. Generally GLUC models are developed to work on areas that are a set of counties within one or more states. The US Census Bureau provides historic population information at the county and census track level and population projections for states to 2025. Such information can be found under http://www.census.gov. By simply apportioning projected population to the area of interest based on the current population, a projection estimate can be established. Local estimates and/or more carefully developed population projection analyses can be conducted.

Economic

Are there any economic projections?

Plans

Zoning

Local zoning plans can be collected to help identify where certain growth will be permitted.

Property Purchases

Planned property purchases designed to change growth patterns can involve outright purchase of property or simply purchase of development rights designed to establish future desired land use patterns. Maps that capture the resulting property development potentials as a result of such purchases can be used to develop GLUC input maps.

Roads and Highways

A major driver of urban growth patterns is the development and upgrade of county roads, state and federal highways and limited access highways such as interstates. Maps capturing alternative plans are essential to the development of the GLUC input maps that provide access to services.

Prepare GLUC Input Maps

The maps described above must be transformed into a set of maps with specific file names and contents. Procedures and scripts for assisting in the transformation are described in Westervelt (2004). The contents of the maps are briefly described below. All maps are raster-gis based, used 30-meter square gridcells, are in an equal-area projection, cover the same ground, and have the same number of rows and columns.

Overview of Maps

Boundary

This map simply indicates the boundaries of GLUC simulation runs. Each grid cell in which the model will be run is indicated with the number 1 and all other cells are given the value of zero.

Metrobuffer

Cells defined as within a growth sphere of influence are indicated with a 1, others are zero. This map can capture the extent of the anticipated legal boundaries of cities over the course of a GLUC simulation.

Nogrowth

The nogrowth map is also a one-zero map. Cells that cannot grow are indicated with the value of 1. In the sample map below, the cells coded with a 1 are associated with interstate highway rights of way. Other examples of no-growth cells are those associated with water, no-growth zones, parks, forests, nature preserves, etc. This map is used to capture many alternative land management policies (e.g. zoning) and investments (e.g. purchase of property and/or property rights.)

Landcover

The National Land Cover Data (NLCD) classification system is used. The raster map showing color-coded categories below use the NLCD category definitions found in the following table.

The NLCD category numbers represent landcover as follows:

11. Open Water	42. Evergreen Forest
12. Perennial Ice/Snow	43. Mixed Forest
21. Low Intensity Residential	51. Shrubland
22. High Intensity Residential	61. Orchards/Vineyards/Other
23. Commercial/Industrial/Transportation	71. Grasslands/Herbaceous
31. Bare Rock/Sand/Clay	81. Pasture/Hay
32. Quarries/Strip Mines/Gravel Pits	82. Row Crops
33. Transitional	83. Small Grains
41. Deciduous Forest	84. Fallow
	85. Urban/Recreational Grasses

Further details on these definitions can be found at:
http://landcover.usgs.gov/classes.html

Slope

The slope map indicates the slope of the cell in degrees

Forest_attractor

The forest attractor is simply the straight-line distance to the nearest cell identified as having forest in the land cover map. Cells containing forest are coded as 0 (zero distance to the nearest forest). The cell size of the map below is 30-meters and the distances to nearby cells containing forest is obviously recognizing this cell size.

Water_attractor

The water attractor map is identical in concept to the forest attractor. In the sample map portion displayed here there is a body of water above the area just to the left of center.

Highway_attractor

This and the following three maps identify the driving time in minutes to the nearest state highway, intersections of open roads, limited-access highway ramps and main roads. The creation of these maps relies on the development of a cell-traversal travel time map. Then, GIS analyses that identify the minimum travel time to the respective resources are conducted to create these four maps. In the images below, black represents zero travel time and the whiter the cell, the greater the travel time. The pure white areas indicate no-growth locations and hence.

Intersection_attractor

Driving time in minutes to enter the nearest intersection. An intersection represents an opportunity to go in a completely different direction and assumes that proximity to intersections is attractive.

Ramp_attractor

Driving time in minutes to enter the nearest limited access highway (e.g. interstate highway).

Road_attractor

Driving time in minutes to the nearest main road.

Cities_attractor

The cities attractor map denotes driving time to the nearest employment centers (municipalities). Because there are many municipalities in a study area, they are grouped into a few categories by population (e.g. large, medium and small cities). Attractor map is created for each group of municipalities by the same way as road attractor maps. Attractor map for each group is given different weights.

General Approach for Creating GLUC Input Maps

The GLUC modeler must develop the input maps (shown above) for their area of interest. There are many approaches that can be used and users are encouraged to explore options and opportunities. However, the following figure suggests the maps that might be used to create the GLUC input maps (identified along the right side). The grey-background maps along the left side are locally specific user maps that capture plans that GLUC will test

with respect to anticipated urban growth. Those without the grey background represent the generic nationally available maps. Two temporary maps (bottom center) are suggested as important steps in the process of creating attractiveness maps that consider driving time on the current and planned road/highway network. One is simply a description of the driving time required to cross each 30-meter square cell and is used to generate several needed images. The other indicates the fundamental economic draw of cities and their associated job and commercial centers.

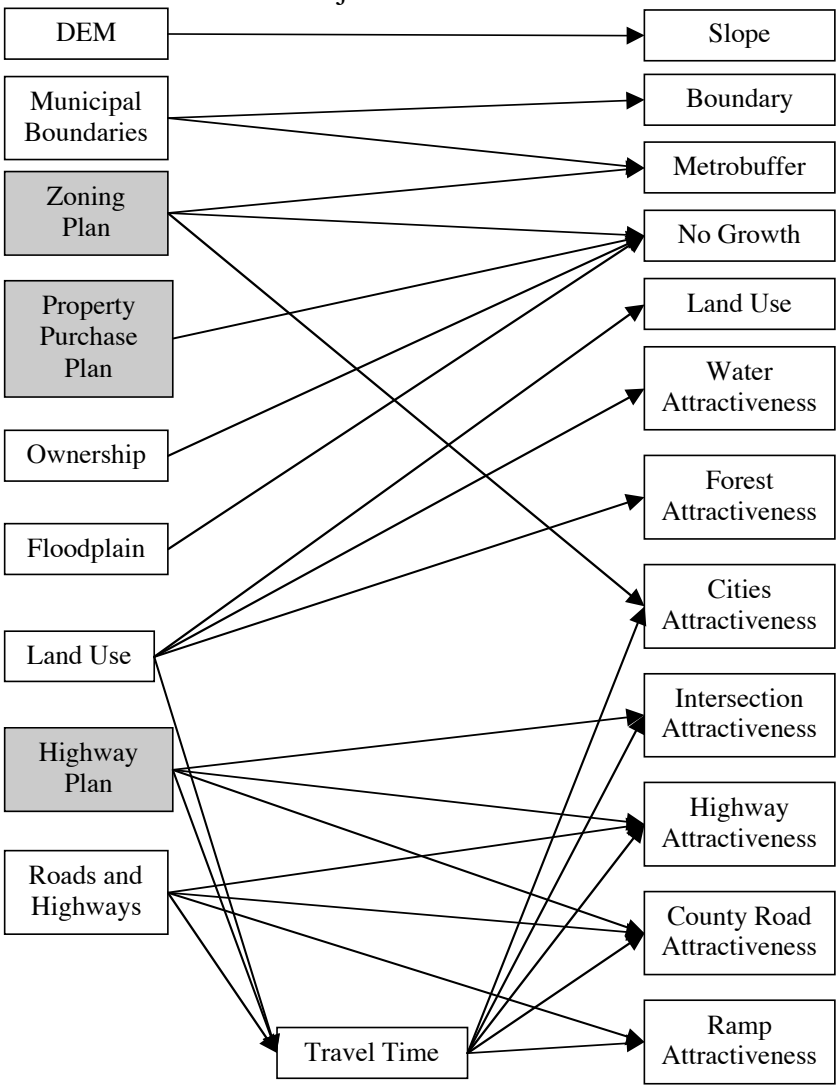

Figure 1: Converting Raw Maps to GLUC Input Maps

Most of the map processing is straightforward. Creation of the slope map requires running standard slope-aspect analysis models available in most raster GIS packages. The boundary map is simply an overlay identifying the counties involved in the simulation process. Similarly, the Metrobuffer map is a 1-0 map identifying the extent of anticipated edges of cities through the course of a simulation. These, and the no-growth map are most challenging with respect to extracting agreement from the user community. Forest and water attractiveness are simply proximity analyses that give distance from each cell to the closest cell containing forest or water.

The road, highway, ramp, and intersection attractiveness maps are based on the travel time to drive to these features. A travel time map is first developed, which identifies the time (in minutes) required to traverse each cell. Artificially high values are assigned to such areas as water, permanent easements for limited access roads/highways, and railroads to essentially indicate that passage is impossible. Using a raster GIS cumulative cost function, this map can be used to generate the various travel-time based attractiveness maps.

The most challenging, and perhaps the most important, input map to generate is the cities_attractor map. This map captures the notion of travel time to jobs and shopping.

Setup GLUC

Acquire GLUC Configuration Files

GLUC must be installed and working on your Unix-based computer at this point. Sample GLUC configuration file information can be downloaded by starting at the SERM web site at: http://www.cecer.army.mil/KD/SERM. Follow the links here to GLUC where the GLUC model, associated configuration file, and installation data files will be found. These are available as compressed archive files. If you are developing a completely new GLUC model, use an available working configuration file as an example.

Create a GLUC Project Under GLUC

Begin by choosing a location in your system's directory structure that has plenty of room. GLUC map input and output files take up the most space. Models can easily involve 10-100 megabyte map files. With about a dozen input maps and at least two output maps for each of many runs, you should select a location with the ability to hold at least 5 gigabytes of data.

GLUC works with a specific arrangement of files and directories (Figure 2 and Appendix A). The project directory is the location where the GLUC file structure will be built to hold all associated files in a specific arrangement. The project is essentially the location for which the GLUC model is being built.

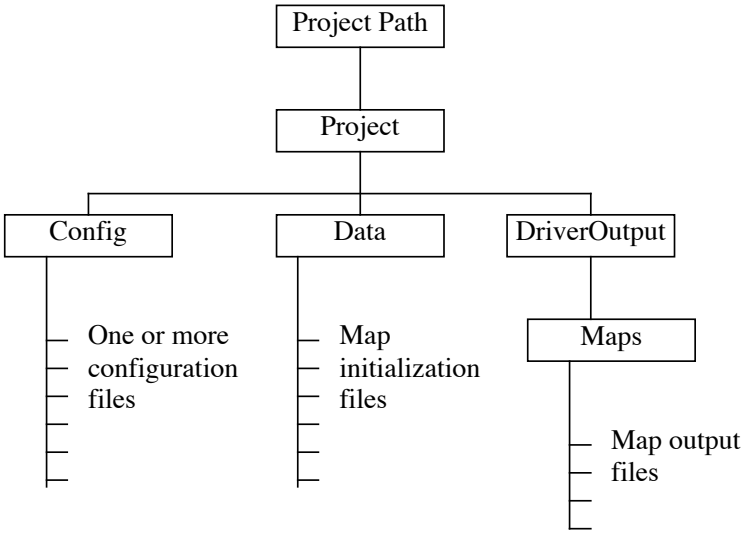

Figure 2: GLUC Model Directory Structure

To begin, you must create the directory structure and place an appropriate configuration file and input map files into the Config and Data directories.

Install Input Data

You must place your raster GIS data files into the GLUC model directory structure so that the model configuration file can point the model to them. GLUC models can, depending on how GLUC was configured and built on your machine, read GIS map data from different types of files. The instructions below are for MAP II GIS data files. MAP II was a Macintosh based GIS system which may no longer be available. The map format however is relatively generic. Each map consists of two files: a binary map file holding integer categories using one to four bytes per grid cell (GLUC only uses one and four-byte files), and a header file identifying the binary file, the number of rows and columns in the map and the number of bytes per cell. A sample header file looks like this:

```
FILETYPE=INTERCHANGE
ROWS=4045
COLUMNS=4810
CELLSIZE=1
UNITS=M
FORMAT=BIN
SIZE=4
LOCATION=″fb_forest_att.bin″
```

The only differences in header files from map to map will be the ROWS, COLUMNS, SIZE, and LOCATION. And, maps being used in any single GLUC run must have the same number of ROWS and COLUMNS. The LOCATION is the name of the corresponding binary map. An excellent double check of the file is to multiply the ROWS, COLUMNS, and SIZE values and compare with the actual byte size of the file

identified in LOCATION. Most raster GIS software systems have the ability to generate the binary file,, especially for 1-byte per cell files.

Install and Edit Configuration File

The final step to be completed before running a GLUC model is to edit the configuration file. If you are running a downloaded model you will have a configuration file that matches the input data files and it will be best to start with it. Place the downloaded data files in the Data folder and copy the configuration file to the Config directory under your project.

The configuration file is in ASCII format and is human editable and provides access to all of the variables associated with a GLUC model. You can easily reset any variable to a fixed value or to values provided in the GIS maps. You can also arrange to have variables displayed during a simulation run and/or stored on disk for later analysis. To provide a head start, here are some translations of some of the configuration file entries:

"Global" information is provided on a line that begins with a "#" symbol. Example:

```
# global  DS(1.0,0) n(1) s(4332) ngl(1) op(0) OT(1,0,38) d(0) UTM(0,0.0,0.0)
```

The information currently used from this line is as follows:

- s(4332) - the "s" indicates that the following argument is the seed for the random number generator
- OT(1,0,38) – "OT" introduces the time between model time steps, the start time, and the ending time
- d(0) – "d" is used to identify the level of debug information. The higher the value, the more debugging. Generally users should use "0".

The remainder of the information is currently ignored, but may be used in the future.

"Module" parameters are introduced with a line that begins with the dollar ($) sign. Example:

```
$ my_module   g(M, boundary.m2, default) AL(0,0) d(0)
```

This can be deciphered as follows:

- The initial string is the module name
- g(M, boundary.m2, default) – the first and third arguments are ignored. The second points to the MapII file that contains the map showing which cells will be active in the simulation.

The remainder of this line is currently ignored.

Most of the configuration file contains variable information and these lines each begin with an asterix (*). Many, if not all of the variables used in the GLUC model are listed and it is best to begin with a preexisting configuration file. Excerpts below provide a look at the argument options:

```
* BUFFER_MAP                    c(M, cityBuff.m2)
* CITIES_ATTRACTOR_MAP          d(M, cities_att.m2)
```

* FINAL_LAND_USE	M(M, 1, summary)
* LAND_USE	s(1) sC(C) M(M, 10, base)
* DICE	r(s)
* NEW_COM_PROJECTED	ft(u)
* OPEN_INIT	pm(62738)

Only a few of the arguments are currently used as follows:

- pm(1234) – This indicates that the variable is a fixed parameter and will be set with the associated number.
- c(M, map.m2) and d(M, map.m2) – Indicates that the variable is to be set from maps found in the "Data" directory under the project with the associated map name. The "M" argument indicates a MapII format file, but currently no other formats are supported. The "c" and "d" versions are currently treated identically.
- M(M, 10, base) – This expression indicates that the associated variable is to be written out in MapII format. The second argument indicates the scale and the third is the map name.

The remainder of the arguments is currently ignored.

Calibrate

To be written.

Run GLUC

Running GLUC, assuming all has gone well with the steps leading up to this point, is a simple matter using the command "gluc". Entering "gluc –help" results in the following:

```
Usage: ./gluc [SME options] [options]

SME options:
 -ppath <path> : sets the project path
 -p <project>  : sets the project name (directory under project path)
 -ci <config>  : specify the configuration file

Options:
 --newk        : turns on improved K factor (default)
 --oldk        : turns on the old version of K factor
 --resoff      : turns off residential development
 --comoff      : turns off commercial development
 --osoff       : turns off openspace development
 --eqrows      : distribute rows equally across processors
 --eqcells     : distribute cells equally across processors
 -r || --random : randomly seeds random number generator
 -f || --final  : generates only final landuse and change map
 -d || --debug  : turns on debugging information
 -h || --help   : display this information
```

Using command line arguments you point gluc to the location of projects (a directory), your project (a directory in the location directory), and a configuration file that will be found in the Config directory under your project.

Review Results

After running, any results you've asked for (via the configuration file) will be in the folder:

../DataOutput/Maps

These are in the format you requested (MapII is recommended) and you can import them into your GIS for viewing and post-processing and analysis. The MapII format is simple. Each map is associated with two files – a header file and a map file. The header file contains ascii information about the number of rows and columns, the number of bytes per pixel, and the name of the map file. The map file is a binary file containing integers that is exactly the number of rows times the number of columns times the number of bytes per pixel long (in bytes). For example a 100 row by 100 column map of pixels that have values encoded in 2 bytes per pixel is 100 * 100 * 2 = 20000 bytes in size.

One way to view the maps is to turn them into more standard formats using the PNM image processing suite (Unix freeware). The following script can be used to assist in the conversion to GIF format files:

```
#!/bin/csh -f

# map2gif -
# Convert MapII file to GIF using a netpbm pipeline.  Images are created
# at the full resolution of the MapII file.

if ($#argv == 0) then
   echo Usage: $argv[0] <MapIIfile> [<outfile>] [<palette>]
   echo <outfile>: defaults to the MapII file with .gif as a replacements
   echo           extension.
   echo <palette>: defaults to mLEAM standard palette
endif

set fin=${1}
if ( ! -r $fin ) then
    echo MapII file $fin does not exist or is unreadable
    exit(-1)
endif

# set output file from command line or create from input file
if ($#argv > 1) then
    set fout = ${2}
else
    set fout = $fin:r.gif
endif

# set palette file from command line or use default (ugly)
if ($#argv > 2) then
    set pal = ${3}
```

```
else
    set pal = ./mleampal.ppm
endif

# parse the values from the header file
set x = `sed -ne 's/^COLUMNS=//p' ${fin}`
set y = `sed -ne 's/^ROWS=//p' ${fin}`
set bytes = `sed -ne 's/^SIZE=//p' ${fin}`
set loc = `sed -ne 's/^LOCATION=//p' ${fin} | tr -d "\307\310"`

# WARNING: we should should do something clever about working
# with the number of bytes per cell.  We currently ignore this
# and assume 1 byte per cell.

# if path given prepend to binary file
if ($fin:h != $fin:t) then
    set loc = $fin:h/$loc
endif

# create the binary file if necessary
if ( ! -r $pal ) then
set pal = ./mleampal.ppm
uudecode << 'PAL'
begin 640 mleampal.ppm
M4#8*,C4V(#$*,C4U"LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9C/,_\R99LR99LR99LR99LR99LR99LR99LR99LR99O__9O^9,\PS`````,R9
M9LR99LR99LR99LR99LR99LS,_\QF_\QFF<R99LR99LR99LR99LR99LR99LR9
M9C/,`#-F,S.9`,R99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9IG,F<R99LR99LR99LR99LR99LR99LR99LR99LS,F9EF,YF99LR99LS,
M9LR99LR99LR99LR99LR99F;,F6;,S,R99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
M9LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR99LR9
09LR99LR99LR99LR99LR99LR9
`
end
'PAL'
endif

# use netpbm tools to covert binary file to gif
rawtopgm ${x} ${y} ${loc} | pgmtoppm -map $pal | ppmtogif > $fout
#rawtopgm ${x} ${y} ${loc} |  ppmtogif > $fout
```

Running GLUC on Super Computers

It is assumed that you, the reader of this section:

- have successfully run GLUC on a local single-processor computer
- have acquired a login to a HPC machine on which GLUC has been installed and compiled
- are familiar with the local Mass Storage Facility (MSF)
- are familiar with the local Load Share Facility (LSF)

At the time this was written, the information was correct for running GLUC on the HPC MSRC located at the ERDC Information Technology Laboratory (ITL) in Vicksburg, Mississippi.

GLUC, when run on large areas requires the use of high performance computers. The Department of Defense High Performance Computing Initiative funded the porting of GLUC to a number of computers. Binary GLUC executables for these computers are available at the GLUC Web site.

ERDC MSRC HPC-GLUC Tutorial

These instructions are for the COMPAQ (emerald) and SGI Origin (ruby) computers at the ERDC MSRC. As described above, running GLUC involves bringing MapII format input data maps and a configuration file together within a specific file structure and then invoking a compiled version of gluc with arguments that point to this information. For the COMPAQ and SGI Origin machines at the MSRC, sample configuration and data files are available on the associated mass storage device along with a csh script that automates the entire process of running gluc on the sample data and scenarios.

Before running the software you must have a login to the machines and be familiar with the procedures for submitting, monitoring, and retrieving results of HPC jobs.

The following simple steps provide a way for you to load and run the LEAM GLUC program on a variety of pre-developed data based with pre-developed configuration files. Once you are logged in, the first step is to get a copy of the script, mLEAM.csh, into your directory using the command:

```
archive get -C /erdc2/jefft mLEAM.csh
```

A version of this script is found in Appendix B, but continual developments and refinements require you to retrieve a copy of the latest version. Next, edit the script and follow the internal instructions. There are several decisions that you must capture in your edits:

1. Choose your request for HPC resources that includes number of processors, amount of memory, maximum amount of run time, etc.
2. Choose a directory name for running the program and capturing the results

3. Choose a sample configuration (and, indirectly, sample data)

Finally you can run the script. The GLUC directory structure is automatically created and initialized in your work directory (e.g. /Work/your-login-name) with data and configuration file. The gluc program is run with appropriate arguments and the results may be found in the directory structure under "DriverOutput". Inspection of the mLEAM.csh script will help you develop your own data sets and run gluc using information in the previous sections of this document.

Further information about the LEAM GLUC software, manuals, data, and configuration files may be found at http://www.cecer.army.mil/KD/SERM.

Appendix A: GLUC's Files

Directory/File	Notes
./Config:	
base.conf	User editable gluc configuration files
high.conf	
low.conf	
uber.test	
./Data:	
boundary.bin	Raster map files. The .m2 files are short ascii header files for the .bin files. The .bin files have a byte size equal to the rows times columns * bytes per cell. Actual file names are specified in the configuration file
boundary.m2	
cities_att.bin	
cities_att.m2	
forest_att.bin	
forest_att.m2	
highway_att.bin	
highway_att.m2	
intersection_att.bin	
intersection_att.m2	
landcover.bin	
landcover.m2	
metrobuffer.bin	
metrobuffer.m2	
nogrowth.bin	
nogrowth.m2	
ramp_att.bin	
ramp_att.m2	
road_att.bin	
road_att.m2	
slope.bin	
slope.m2	
water_att.bin	
water_att.m2	
./DriverOutput:	
/Maps	
summary.bin	Result maps in .m2 format. Actual names are specified in the configuration file.
summary.m2	
change.bin	
change.m2	

Appendix B: Sample GLUC Execution Script

Sample csh script for submitting GLUC runs on the ERDC MSRC SGI Origin 3000 (ruby) and the Compaq SC45 (emerald):

```
#!/bin/csh

## Last Modified: Mon Oct  6 11:26:08 CDT 2003
#########################################################################
#########################################################################
# This script was specifically prepared to support the October 2004
# Beta Test of the mLEAM software running on two HPC machines at
# the ERDC MSRC.  The effort is supported by the DoD's HPCMP through
# the CHSSI program.
#
# Information about mLEAM can be found through:
#  http://www.cecer.army.mil/KD/SERM (click on mLEAM)
#
# In the top part of this script, set variables as instructed.  The
# bottom part uses these variables to run mLEAM.
#
# The latest copy of this script can be retrieved with the
# following command from any ERDC MSRC machine:
#   archive get -C /erdc2/jefft mLEAM.csh


#########################################################################
#########################################################################
# If you are running on a Compaq or Origin, modify the following
#########################################################################
# Modify these Load Sharing Facility (LSF) options as necessary:
#  account (-P)
#  number of processors (-n)
#  memory in kilobytes (-M)
#  maximum wall clock time in minutes (-W)
#  job name (-J)
#  output file (-o)
# By convention, set the job name (-J) and output file (-o)
# to match the project name.
#BSUB -P erdcvenq
#BSUB -n 2
#BSUB -M 8000000
#BSUB -W 180
#BSUB -o sample-1.o
#BSUB -J sample-1

# If you are running under LoadLeveler (NAVO IBM, etc) set these...
#########################################################################
# Modify these LoadLeveler options as necessary:
#  account_no
#  number of processors (-r)
```

```
#  Queue (-q)
#  maximum Cray MPP processing elements (-l mpp_p)
#  maximum per-process CRAY wall-clock residency time (-l p_mpp_t)
#  maximum per-request CRAY wall-clock residency time (-l mpp_t)
# @ shell = /bin/csh
# @ account_no = ERDCVenq
# @ environment = ENVIRONMENT=BATCH; COPY_ALL
# @ output = leam1.out
# @ error  = leam1.error
# @ network.MPI = csss,not_shared,US
# @ job_type = parallel
# @ job_name = leam1
# @ node = 4
# @ total_tasks = 32
# @ node_usage = not_shared
# @ wall_clock_limit = 5:00:00
# @ class = batch
# @ queue


######################################################################
# Number of processors to be used during the run.  This number must match
# the "#BSUB -n" argument above.
#
setenv MP_SET_NUMTHREADS 32

######################################################################
# Set the project name.  Project name will be used to create the
# current working directory under ${WORKDIR}.  If ${WORKDIR}/${project}
# exists files may be overwritten so it's best to choose a unique
# project name for each model run.
set project=sample-1

######################################################################
# Set the configuration file name by uncommenting one of the lines below.
# Each configuration file describes a specific scenario.
# Associated data files are also automatically identified and retrieved
# based on the name of the configuration.
# (e.g. sample_base.conf is associated with sample.data.tar)
#
#### Small sample scenarios to start out your experimentation
#set config=sample_base.conf
set config=sample_xmdf.conf
#set config=small_3p.conf
#set config=small_10p.conf

#### Army Installation Scenarios
#------- Fort Benning, Georgia
#set config=fbenning_base.conf
#set config=fbenning_high.conf
#set config=fbenning_low.conf
#set config=fbenning_uber.conf
#set config=fbenning_xmdf.conf
```

```
#------- Fort Bragg, North Carolina
#set config=fbragg_base.conf
#set config=fbragg_high.conf
#set config=fbragg_low.conf
#set config=fbragg_uber.conf
#set config=fbragg_xmdf.conf
#------- Camp Ripley, Minnesota
#set config=cripley_base.conf
#set config=cripley_high.conf
#set config=cripley_low.conf
#set config=cripley_uber.conf
#------- Fort Carson, Colorado
#set config=fcarson_base.conf
#set config=fcarson_high.conf
#set config=fcarson_low.conf
#set config=fcarson_uber.conf

#### These are scenarios for St Louis, MO and East St Louis, Illinois
#------- Standard runs with low, medium, high, and very high (uber)
#        population projections
#set config=stlme_base.conf
#set config=stlme_high.conf
#set config=stlme_low.conf
#set config=stlme_uber.conf

#### These are scenarios for Peoria, Illinois
#------- Standard runs with low, medium, high, and very high (uber)
#        population projections
#set config=tc_base.conf
#set config=tc_high.conf
#set config=tc_low.conf
#set config=tc_uber.conf
#------- Agriculture preservation scenarios
#set config=tc_agpreserv_base.conf
#set config=tc_agpreserv_high.conf
#set config=tc_agpreserv_low.conf
#------- River bluff protection scenarios
#set config=tc_bluff_base.conf
#set config=tc_bluff_high.conf
#set config=tc_bluff_low.conf
#------- Facility Planning Areas (FPA) scenarios
#set config=tc_fpa_base.conf
#set config=tc_fpa_high.conf
#set config=tc_fpa_low.conf
#set config=tc_grcell_base.conf
#set config=tc_grcell_high.conf
#set config=tc_grcell_low.conf
#------- An interstate bypass scenario
#set config=tc-ringroad_base.conf
#set config=tc-ringroad_high.conf
#set config=tc-ringroad_low.conf
#------- A proposed State highway scenario
#set config=tc-sr29_base.conf
```

```
#set config=tc-sr29_high.conf
#set config=tc-sr29_low.conf

########################################################################
# Set the archive variable if you wish to automatically copy
# the new map files to the mass storage system after the
# application completes.  Files will be stored in a tar file
# called $(project).results.tar

#set archive=

########################################################################
########################################################################
# ----- The following lines should not need to be changed -----

# Set the machine architecture where the batch job will run.
# This allows the correct executable to be retrieved.  Possible
# options for this variable are currently 'IRIX64', 'OSF1', and
# 'AIX' (IBM Power4).

set arch=`uname`

# Set the base name of the program to be executed.  Generally
# this only needs to be set when testing new executables.  If
# the program variable is not set then it is automatically
# set by extracting it from the configuration file.

set program=glucx

# Set the input data tar file name.  This can generally be
# guessed from the configuration file name provided above.  Because
# configuration files and data files must correspond, care
# must be taken when overriding this variable.

#set input=

# Set the default mass storage directory.  Everything is
# currently stored in /erdc2/jefft

set mssdir=/erdc2/jefft

########################################################################
########################################################################
# ----- DO NOT EDIT BELOW THIS LINE -----

# Set Runtime Environment
setenv MPI_TYPE_MAX 8192

# Build the SME Runtime Directory Structure
set wdir=${WORKDIR}/${project}
mkdir ${wdir}
cd ${wdir}
mkdir Config Data Driver DriverOutput DriverOutput/Maps
```

```
# Retrieve the Configuration File
cd ${wdir}/Config
msfget ${mssdir}/${config}

# Retrieve the Input Data
cd ${wdir}/Data
if ( $?input == 0 ) then
   set input=`echo $config | sed -e 's/[_.].*//'`
endif
msfget ${mssdir}/${input}.data.tar
tar xf ${input}.data.tar
rm ${input}.data.tar

# Retrieve and Run the Application
cd ${wdir}/Driver
if ( $?program == 0 ) then
    set program=`echo $config | sed -e 's/[_.-].*//'`
endif

set args=(-f -d --newk  -ppath ${WORKDIR} -p ${project} -m leam -ci ${config})
switch ( $arch )
    case IRIX64:
        msfget ${mssdir}/${program}.IRIX64 ${program}
        chmod a+rx ${program}
        time mpirun -np ${MP_SET_NUMTHREADS} ./${program} ${args}
        breaksw
    case OSF1:
        msfget ${mssdir}/${program}.OSF1 ${program}
        chmod a+rx ${program}
        time prun -n ${MP_SET_NUMTHREADS} ./${program} ${args}
        breaksw
    case AIX:
        breaksw
    default:
        echo "ERROR: unknown architecture ${arch}"
        exit
endsw

# Preserve the Results
cd ${wdir}/DriverOutput/Maps
if ( $?archive ) then
    tar cf ${project}.results.tar *
    msfput ${archive}/${project}.results.tar
endif
```